# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| MAY 2010 | Conference Paper Postprint | January 2009 – January 2010 |

**4. TITLE AND SUBTITLE**

ONTOLOGY MATCHING ACROSS DOMAINS

**5a. CONTRACT NUMBER**
FA8750-09-C-0058

**5b. GRANT NUMBER**
N/A

**5c. PROGRAM ELEMENT NUMBER**
62702F

**6. AUTHOR(S)**

Renato Levy, Jakob Henriksson, Margaret Lyell, Xiong Liu, and Michael J. Mayhew

**5d. PROJECT NUMBER**
558E

**5e. TASK NUMBER**
9B

**5f. WORK UNIT NUMBER**
03

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Intelligent Automation, Inc.
15400 Calhoun Drive, Suite 400
Rockville, MD 20855

**8. PERFORMING ORGANIZATION REPORT NUMBER**
N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFRL/RIEB
525 Brooks Road
Rome NY 13441-4505

**10. SPONSOR/MONITOR'S ACRONYM(S)**
N/A

**11. SPONSORING/MONITORING AGENCY REPORT NUMBER**
AFRL-RI-RS-TP-2010-25

**12. DISTRIBUTION AVAILABILITY STATEMENT**
Approved For Public Release; Distribution Unlimited. PA #: 88ABW-2010-0886
Date Cleared: 1-March-2010

**13. SUPPLEMENTARY NOTES**
This paper was accepted for publication in the proceedings of the Ninth International Conference on Autonomous Agents and Multi Agent System Interoperability Workshop, Toronto, CA; May 10-14, 2010. If published, this work may copyrighted. One or more of the authors is a U.S. Government employee working within the scope of their Government job; therefore, the U.S. Government is joint owner of the work and has the right to copy, distribute, and use the work.

**14. ABSTRACT**
Ontologies are often used to annotate information (metadata) that is passed between domains during negotiation. In that sense, Ontology matching is critical for the receiving domain to gather the correct meaning of the data, and hence critical for interoperability. Many Ontology matching algorithms have been proposed in the literature but in general they all assume that there is a considerable amount of knowledge about both ontologies (sender and recipient). This assumption is not true in many cases. In this paper, we present an approach that does not require such assumption, allowing the parts to keep a considerable amount of secrecy on their Ontology while still providing the required matching functionality.

**15. SUBJECT TERMS**
Cross-Domain Ontology Matching, Ontologies; Metadata Policies; Ontology Metrics

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Michael J. Mayhew |
| U | U | U | UU | 14 | 19b. TELEPHONE NUMBER (*Include area code*) N/A |

# Ontology Matching across Domains

Renato Levy[1], Jakob Henriksson[1], Margaret Lyell[1], Xiong Liu[1], Michael J. Mayhew[2]

[1] Intelligent Automation, Inc., 15400 Calhoun Drive, Suite 400
Rockville, MD, USA, 20855, +1-301-294-5200
{ rlevy | jhenriksson | mlyell | xliu }@i-a-i.com
[1]

[2] Air Force Research Lab, Information Directorate
525 Brooks Rd, Rome, NY 13441 Michael.Mayhew@rl.af.mil

**Abstract.** Ontologies are often used to annotate information (metadata) that is passed between domains during negotiation. In that sense, Ontology matching is critical for the receiving domain to gather the correct meaning of the data, and hence critical for interoperability. Many Ontology matching algorithms have been proposed in the literature but in general they all assume that there is a considerable amount of knowledge about both ontologies (sender and recipient). This assumption is not true in many cases. In this paper, we present an approach that does not require such assumption, allowing the parts to keep a considerable amount of secrecy on their Ontology while still providing the required matching functionality.

**Keywords:** Ontology, matching algorithm, metadata, cross-domain, multi-language interoperability

## 1 Introduction

Ontologies are often used to annotate information (metadata) that is passed between domains during negotiation. In that sense, Ontology matching is critical for the receiving domain to gather the correct meaning of the data, and hence critical for interoperability. Many Ontology matching algorithms have been proposed in the literature but in general they all assume that there is a considerable amount of knowledge about both ontologies (sender and recipient).

The manner in which an ontology is organized can give valuable insights on the organization's knowledge representation and the importance, complexity or amount of data expressed in this knowledge base. This information in itself is very valuable and the assumption that negotiations can happen across domain boundaries with full disclosure of the domain's ontologies are naïve at best.

In this paper, we present an approach that does not require such assumption, allowing the parts to keep a considerable amount of secrecy on their Ontology while still providing the required matching functionality. In fact, we want, as much as possible, to keep the upkeep of the sending and receiving ontologies separated. This creates an extra layer of complexity for Ontology matching problem, since the metadata associated with the information must be converted to the receiving ontology at the domain boundary.

The ontology matching process across domain boundaries has some extra requirements form the traditional academic problem that makes it unique. Some of these key issues are:

- Multilingual/multicultural → One important issue in the cross-domain arena is that the Ontologies to be matched maybe in different languages (multi-national negotiations), hence syntax proximity is not relevant.
- Independent management/runtime matching → Another important issue to be observed is the ability to handle the matches quickly at runtime, without an extensive preparatory effort, thus allowing the Ontologies to be managed independently.
- Limited information exchange → In the case of cross-domain, the participants of the Ontology matching may not want to disclose their full ontology, but only the necessary information for a correct matching to be performed. One must remember that the need for the ontology matching if often not a translation, but only the adjustment of the metadata and the coherence and continuity of its properties.

Although the existing literature does not directly apply to this practical extended problem, we were able to find relevant work that we believe can be adapted/enhanced to work in our domain.

### 1.1 Related work

Ontology matching is the process of finding semantic correspondence between similar entities of different ontologies. A lot of work has addressed the problem of ontology matching. Here we describe five major matching methods that have been reported in the literature, including graph-based matching, linguistic matching, hybrid matching, learning based matching and probabilistic matching.

1. Graph-based matching or structural matching uses graphs to represent ontologies and computes structural similarities of graphs. Examples of graph-based matching include GMO [1], Anchor-Prompt [2], and Similarity Flooding [3]. GMO is an iterative structural matcher, which uses RDF bipartite graphs to represent ontologies and computes structural similarities between entities by recursively propagating their similarities in the bipartite graphs. This is an approach that we possibly can exploit and hence take a closer look at it in section "Adjacency Matrix-Based Matching Algorithm" below. Anchor-Prompt is an ontology merging and mapping tool, which treats ontologies as directed labeled graphs. The basic idea is that if two pairs of entities are similar and there are paths connecting them, then the entities in these paths are often similar as well. Similarity Flooding is a graph matcher which uses fixpoint computation to determine corresponding nodes in the graphs. The basic idea is that the similarity between two nodes depends on the similarity between their adjacent nodes, or similarities of nodes can propagate to their respective neighbors.
2. Linguistic matching lies in the construction of virtual documents. The virtual document of an entity in an ontology contains the local descriptions as well as neighboring information that contains the meaning of the entity. Then calculating the similarities of entities translates to the problem of calculating document similarities using traditional vector space techniques. V-Doc [4] is an example of linguistic matcher. It exploits the RDF graph to extract the description information from three sorts of neighboring entities, including subject neighbors, predicate neighbors and object neighbors.
3. Hybrid matching uses linguistic information (e.g., name, label, and description) and structural information (e.g., key properties, taxonomic structure) to find correspondences between entities. For example, PROMPT [5] is a hybrid matching tool for user oriented ontology merging. To make the initial suggestions, it uses a measure of linguistic similarity among concept names and mixes it with the structure of the ontology and user's actions. For each operation, it finds conflicts that the operation may introduce and presents new suggestions to the user.
4. Learning based matching is efficient when instances are available in ontologies. GLUE [6] is an example of learning based matching system. It first applies statistical analysis to the available data and uses multiple learners to exploit information in concept instances and taxonomic structure of ontologies. It then uses a probabilistic model to combine results of different learners. Finally it adopts relaxation labeling approach to search for the mapping that best satisfies the domain constraints and the common knowledge.
5. Probabilistic matching is also used on instance level in ontology matching. For example, OMEN [7] is a tool which describes mappings using probabilities and infers new mappings by means of Bayesian Network inference.

The rest of this paper is organized as follows: In section2, we present the overall approach for the extended cross-domain Ontology matching problem, and in section 3 present an example on the matching methodology proposed. Section 4 gives more details on how to successfully implement such methodology. Finally, section 5 discuses our conclusions and the future work in this area.

## 2. Overview of Approach

In a collaborative environment, every participant has their own priorities and perspectives of their reality—they each have their own domain model. The domain model highlights what is considered important and formally structures the domain. Such a domain model is the base of the ontology used to describe the concepts on the domain. It is clear, however, that the ontology of one collaborative participant does not always match the ontology of another participant. In fact, it is highly unlikely that this is the case, while at the same time it is likely that the ontologies overlap to some degree. After all, the participants have a desire to communicate so it's likely they have some overlapping terms in their ontologies. We will later refer to these overlapping terms as anchors.

In a cross domain environment that intends to share information between domains, security plays an important role. Hence, not only do domains have an ontology that describes their world, but they also limitations on how much of this ontology can be shared. These policies refer to the ontology since the policies are specified over the ontology terms (the resources in the domain). We will not focus on the policies here, but it is important to understand that there are properties associated with the ontology terms and policies that limit the amount of information that can be shared.

So, each domain has a domain ontology and security restrictions specified by policies. This means that any data in the domain is classified according to the ontology, and hence has access restrictions in place.

The premise of interoperability is that information (data) can flow between domains and have their key properties recognized, while still being able to ensure the policy restrictions. In short, the premise of interoperability that data requires metadata to be understood, and the realization that there are limitations on how to translate the metadata context greatly increases the overall complexity of the principle. An illustration of the setup is given in Figure 1.

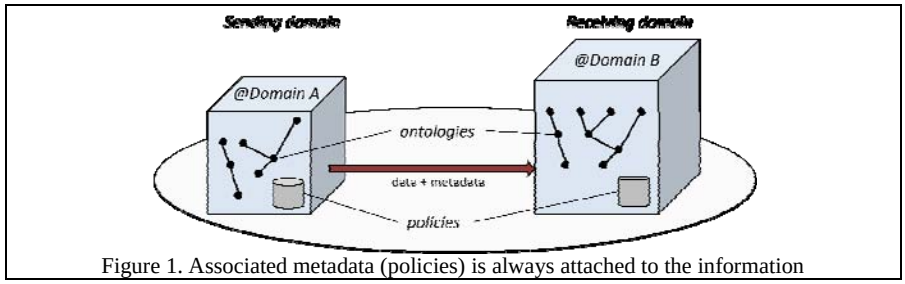

Figure 1. Associated metadata (policies) is always attached to the information

Since the ontologies differ, the security insurances (policies) also do not fit exactly. None the less, the data sent across domain boundaries eventually need to be stored according to the ontology of the receiving domain, and secured by its policies. Hence, to overcome these problems, some issues must be addressed:

- Ontologies cannot be assumed to be fully shared or disclosed between domains, since each domain wants to protect the details of its domain understanding
- Even if we can identify an appropriate concept in the receiving domain ontology that can be used to classify a data item, we cannot assume that the sending domain fully accepts the policy restrictions that are placed on this resource concept.

Even if domains do not want to fully disclose their ontologies, they can agree on certain concepts that they share and can disclose. These concepts would manually be determined by human representatives of the domains. In order to be consistent with nomenclature used in previous literature [8], we call these concepts anchors.

By having a guaranteed partial ontology overlap, it is possible to match a concept in the sending domain ontology with a concept in the receiving domain ontology to a sufficient degree of accuracy. Even if the match is not exact, the sending domain might agree with the security assurances and overall properties provided by the receiving domain and may send the data confident. An initial overview of the process is shown in Figure 2.

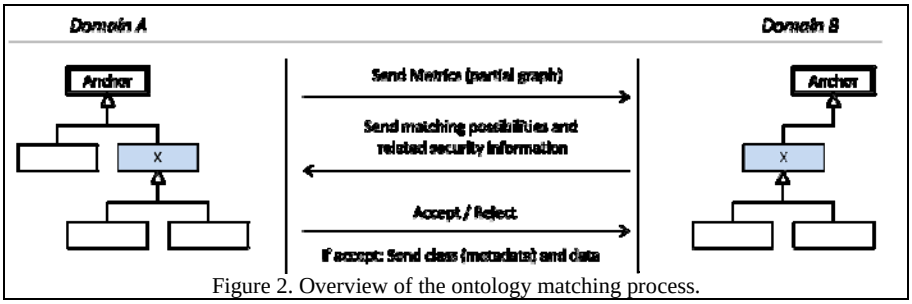

Figure 2. Overview of the ontology matching process.

In more detail, the steps are the following:

1. **Describe matching metrics**. The first step is for the sending domain to get assurances that the data that eventually can be sent with a sufficient matching concept in the receiving domain. Hence, before any data is actually sent, the domains need to negotiate. The data to be sent from the sending domain has some metadata associated with it. In particular, it is declared to be an instance of a particular concept C in the domain ontology. The sending domain, averse to disclosing too much of its ontology, effectively decides on an appropriate subset of the ontology to send over to the receiving domain. This subset is however made anonymous. By this is meant that all entity names, except the anchors (which are already shared), are removed (or changed to meaningless names).

The aim of the sending domain is to provide enough information to the receiving domain such that a similar concept can be found in its ontology by which the received information can eventually be categorized and secured by policies. It should be noted that more than a simple topology of the selected ontology subset is communicated. Rather, what is sent is a set of metrics that can be used for matching against the receiving domain ontology. These metrics are descriptions of how the concept C relates to the anchors. This is important since this is the only way for the receiving domain to be able to find a matching concept (since they share anchors).

An example of a metrics for concept C could be:

(IS-A, 1, “Anchor Concept 1”).

This metric says that the concept C is a subclass of the anchor “Anchor Concept 1”. The metrics:

(IS-A, 2, “Anchor Concept 1”)

for concept C means that C is related to “Anchor Concept 1” by two IS-A (or subclass-of) relationships. That is, there is some concept X that is a subclass of “Anchor Concept 1” and C is a subclass of X. Another metrics that partially describe the sending domain ontology can also be given as we will see in examples below.

**2. Match ontologies.** Once a set of metrics <M1, …, Mn> has arrived at the receiving domain, it tries to determine which concept in its ontology, if any, might be a good match for the data that will arrive. This is done by applying graph search algorithms based on the received metrics. Each entity in the domain ontology is given a score (value between 0 and 1) for each metric. The set of scores for each entity is then combined and normalized into a final value that represents the confidence of it being a good match (again, between 0 and 1). The best k matches are selected and each associated with a key. The reason for using keys is to avoid having to disclose anything of the domain ontology to the sending domain. Then k triples:

<key, relevant properties, matching score>

are sent to the sending domain. This gives the sending domain a chance to pick a desired match.

3. **Metadata and selection.** The sending domain can now make its decision based on the received response triples: the property set for a particular match and its likeliness of being a good match. The most likely scenario is for the sending domain to prioritize on a given property, but this must not be the case. In some cases a good match might be preferred despite detail degradation, while in other cases a lesser ontology match might be preferred when a given property has a higher priority. Nonetheless, the choice lies with the sending domain that is responsible for the data leaving its domain. Without acceptable guarantees given by the receiving domain, the response can also be “reject”, in which case the data is not sent at all. This means that the sending domain does not want to send the data to that particular receiving domain. If the choice instead is “accept”, one of the keys is picked and the data is sent together with the key. The key is here a representative of the metadata in the sense of “data and metadata are inseparable.”

4. **Data storage.** Once the response from the sending domain is received, the receiving domain can classify the newly received data by using the correspondent concept represented by the key.

## 3. Example

In the following we provide an example that demonstrates the intuition behind the steps described above.

Describe matching metrics. The domain ontology for the sending domain is shown in Figure 3. It describes concepts such as “Weather Reading”, “Wind Reading” and “Location”. Notice that we do not only have IS-A relationships, we also have disjoints (e.g. “Wind Reading” and “Hourly Temperature Reading” are disjoint), properties (e.g. location), domain and range restrictions (the domain and range of property “location” is “Weather Reading” and “Location”, respectively), and property restrictions (e.g. “= 1 location”, which means that an instance of “Weather Reading” is related to exactly one instance of “Location” via the property “location”). Hence, we make us of several different kinds of ontology constructs to model our domain.

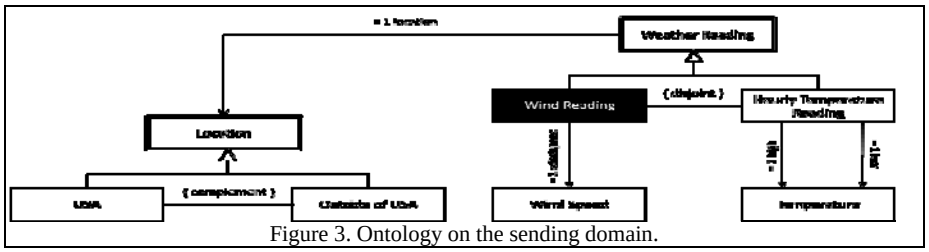


Figure 3. Ontology on the sending domain.

Concepts with a double border are anchors (pre-agreed and shared concepts). In this example the concept “Wind Reading” represents the metadata that is to be sent along with the. Since we know that the receiving domain ontology has the concepts “Weather Reading” and “Location” (they are the anchors), our goal is to describe enough about the concept “Wind Reading” in relation to the anchors such that the receiving domain can do a reasonable match onto its concepts and ontology structure. This description, which we call our metrics, could for example be the ones given in Table 1.

Table 1. Metrics describing a partial ontology

| Metric | Explanation |
|---|---|
| (**IS-A**, 1, “Weather Reading”) | The metadata concept is one step removed from anchor “Weather Reading” via a IS-A relationship. That is, the metadata concept is a subclass of “Weather Reading” |
| (**disjoint**, sibling) | The metadata concept has a disjoint sibling via the IS-A relationship. *Notice that thanks to the first metric (above), we already know that this is in relation to the “Weather Reading” anchor.* |
| (**domain**, 1, = 1 restriction, range: “Location”) | The metadata concept is the domain of an “exactly one value” restricted property that has as range the anchor concept “Location” |
| (**domain**, 1, = 1 restriction, range: *unknown*) | The metadata concept is the domain of an “exactly one value” restricted property that has an unknown range concept. *Again, we know that this description is in relation to the “Weather Reading” anchor, and we know that the range is not an anchor, because otherwise it could be given.* |

It should be noted that all the descriptions of the metadata concept given in Table 1 are in relation to an anchor. This is important since the anchors are the only agreed upon concepts between different domain ontologies. Intuitively, the descriptions in Table 1 correspond to the partial ontology structure highlighted in Figure 4.

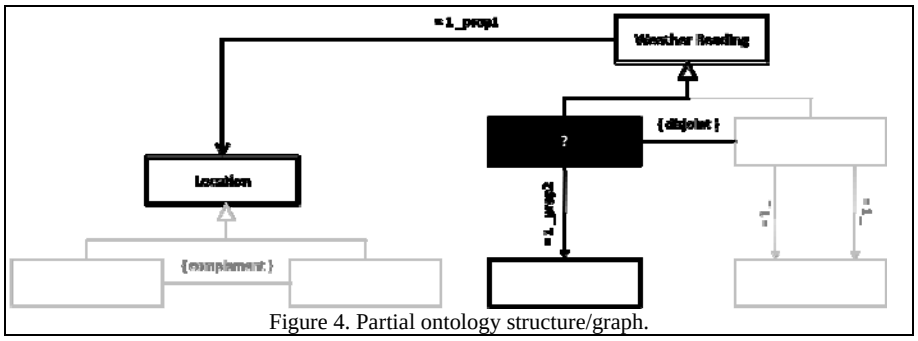

Figure 4. Partial ontology structure/graph.

It should be noted that it is not always desirable to only match the topological structure of ontology graphs. It can also be highly desirable to give larger weight, or preference, to certain kinds of relationships. For example, one could say that a matching IS-A relationship is more important than whether or not a concept is the domain for some specific property. Hence, the relationships between the "nodes" (entities) in the ontology structure can play an important role.

Match ontologies. At the receiving domain, we now assume that the metrics, the partial ontology description, from Table 1 has arrived. The task is now to determine which concept in the local ontology is a likely match for the metadata (concept) that will be sent from the sending domain. The local ontology is illustrated in Figure 5. The ontology describes similar, but different, terms compared to the ontology in Figure 3. That the ontology partly overlap should be clear since they already have agreed on some shared concepts (the anchors).

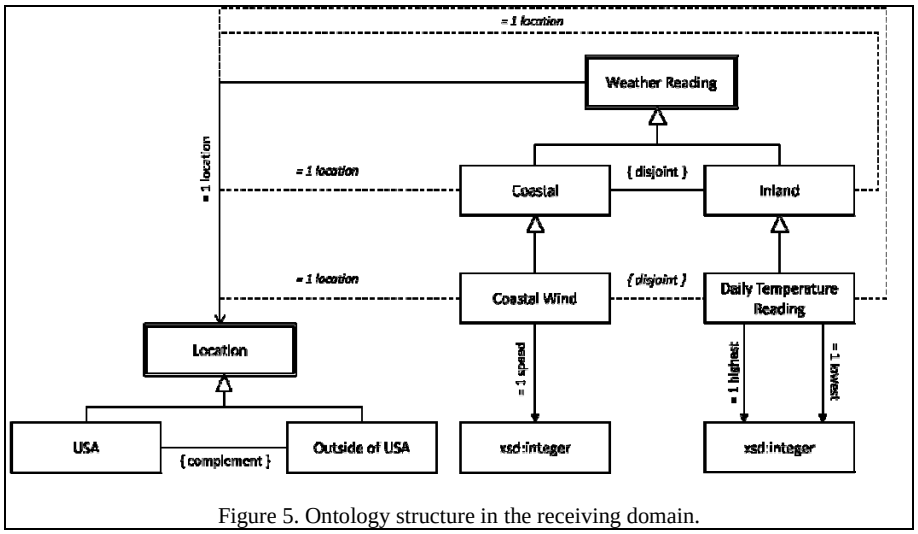

Figure 5. Ontology structure in the receiving domain.

The matching is done by searching the local ontology graph structure and assigning scores to nodes for each of the metrics. An example of scores given to only some of the concepts in the ontology is shown in Figure 6.

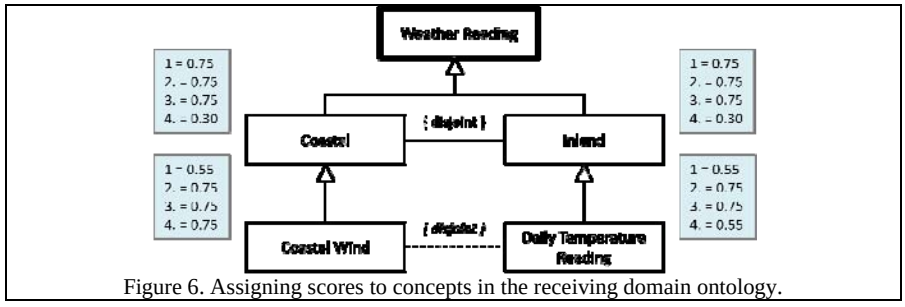

Figure 6. Assigning scores to concepts in the receiving domain ontology.

The orders of the scores correspond to the descriptions in Table 1. For example, the first metric in Table 1 states that the metadata concept is a direct subclass of the anchor "Weather Reading". Both the concepts "Coastal" and "Inland" match this description and get a high score. The concepts "Coastal Wind" and "Daily Temperature Reading" are not exact matches, but close, and receive a lower score. These calculations are done for each of the metrics that are sent from the sending domain. When we add the scores together we get something like what is shown in Table 2. These scores can then be normalized, but this is left out here.

Table 2. Scores for metadata concepts.

| Key | Metadata Concept | Score |
|---|---|---|
| **key1** | Coastal | 0.75 * 0.75 * 0.75 * .30 = 0.1265625 |
| **key2** | Inland | 0.75 * 0.75 * 0.75 * .30 = 0.1265625 |
| **key3** | Coastal Wind | 0.75 * 0.75 * 0.75 * .55 = 0.2320312 |
| **key4** | Daily Temperature Reading | 0.55 * 0.75 * 0.75 * .55 = 0.1701562 |

A cut-off value can be selected to limit the number of choices sent back to the sending domain. For example, 0.16 might be chosen for the above example. Hence, the domain would send back the following choices:

<key3, "property1", 0.23>

<key4, "property2", 0.17>

Here we have assumed that there is some understanding of what the properties description mean, which could be more elaborate and must be pre-agreed between the domains along with the anchor concepts.

**Metadata and selection.** The sending domain looks at the options, decides that the "property1" is equivalent on its original concept for the data to be sent, and decides to go with what is considered the best match. Hence, the sending domain sends the data together with "key3."

**Data storage.** The response is handled in the receiving domain by properly categorizing the data and hence protecting the data according to the policy. In this example, the received data would be tagged with the metadata "Coastal Wind".

**Overview Summary**

The example above has been used to demonstrate how ontology matching can be used to facilitate cross domain security communication. The approach is based on the knowledge that each domain has a domain ontology that acts as its data model. Policies are specified with respect to this data/domain model, which are used to guarantee the security of the underlying data.

For domains to share data (secure cross domain information sharing) they first need to negotiate the terms for sharing the data. This is done by matching their ontologies, but without the requirement to fully disclose their ontologies. To be able to do this, the domains have already agreed to certain common concepts. These fully disclosed and shared concepts are referred to as "anchors."

The domain which is to receive the data, tries to find a good match in its ontology and sends some alternatives back to the sending domain, including information about the properties associated with those ontology matches. The sending domain then has the opportunity to decide what to do, and under which terms to send the data. Once the data is sent, the proper matching can be enforced in the receiving domain, as agreed to by the sending domain.

The crucial point here is to investigate good metrics for describing useful ontology structure with respect to agreed upon ontology anchors. Further to device a successful matching technique that properly can be evaluated and demonstrated to give good results. We outline our initial approach to this matching process below, but more work and evaluation is needed.

## 4. Secure Ontology Matching Algorithm

In this section, we describe our approach to perform graph-based matching of ontologies. Unlike traditional ontology matching which matches the entities between different ontologies, our problem is to find the best match (i.e., an entity) in the receiving ontology for a given metadata in the sending ontology. Also, only graph-based matching (or structural matching) is allowed in our problem, because the descriptions of the entities in the ontology are not to be shared.

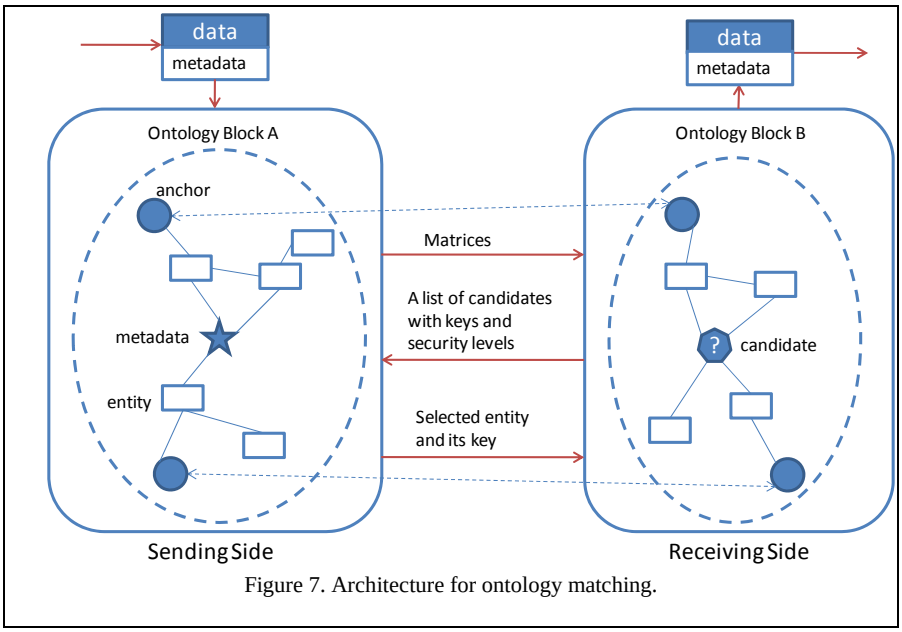


Figure 7. Architecture for ontology matching.

**Architecture Overview**

Figure 7 shows the architecture for ontology matching. On each side, there is a domain ontology (or ontology block). Also, there are anchors that are predefined by humans. Anchors are alignments with high similarities. They are necessary for finding matches of the metadata. The red arrows show the information flow. When the sending side receives a metadata, it will construct matrices or a set of descriptions describing the relationships between the metadata and anchors. The matrices are then sent to the receiving side, which is responsible for finding a list of candidates. Since the contents of the candidates are not allowed to be shared, the receiving side will only send a list of candidate keys with their properties to the sending side. Then the sending side will pick a candidate and send back its choice. Finally the receiving side will attach the selected entity (i.e., metadata) to the data and forward it.

The major processing steps on the sending side include:

1. If the ontology is a large ontology with more than 1000 entities, partition it into blocks of RDF triples. The divide-and-conquer method described in [8] will be used for partitioning here.
2. Given a metadata M, retrieve the ontology block and construct a graph G for the block.
3. Apply a depth-first search algorithm to construct the matrices or a set of descriptions describing the position of the metadata with respect to the anchors in the block. (Note: refer to the algorithm design part for the details of the depth-first search algorithm)
4. Send the metrics to the receiving side.

The major processing steps on the receiving side include:

1. If the ontology is a large ontology with more than 1000 entities, partition it into blocks in the same way as on the sending side.
2. For each metric obtained from the sending side, extract the information about anchor. Retrieve the ontology blocks that contain the anchor, and construct graphs for the blocks.
3. For each metric, apply a width-first search algorithm to compute the scores of candidate entities that could potentially match the metadata. Note that this search is done in every retrieved ontology block. (Note: refer to the algorithm design part for the details of the width-first search algorithm)
4. Compute the weighted sums of scores for the candidate entities. Rank the candidates and attach information about key and security level.
5. Send the ranked list of candidate entities to the sending side.

**Search-based Matching Algorithm**

In this section, we describe the algorithm design for the matching process. On the sending side, we will design a depth-first search algorithm to construct the metrics. On the receiving side, we will design a width-first search algorithm to generate a list of candidate entities.

**Error! Reference source not found.** shows the pseudo code for constructing the matrices on the sending side. This depth-first search algorithm takes the ontology graph G and the metadata M (a vertex of G) as input. It assumes that there is a list of anchors in the ontology (Line 3). It initializes a tree T to the starting vertex, and a list L which stores the edges that are visited (Line 4, 5). In the search function *Search(vertex v)*, the algorithm first marks the vertex as visited and checks if the vertex is an anchor or not. If the vertex is an anchor, the search stops (Line 21, 22, 23). If a vertex v has several unmarked neighbors, the edges between the vertex and the neighbors will be appended to the list L (Line 24, 25). Note that it would be equally correct to visit the neighbors in any order. The easiest method to implement would be to visit the neighbors in the order they are stored in the adjacency list for v. As a depth first search algorithm, it removes edges from end of list L so that the list acts as a stack rather than a queue (Line 10). Also, each vertex is clearly marked at most once, each edge is added to the list L at most once, and therefore removed from the list at most once. The spanning tree T constructed by the algorithm is a depth first search tree, where the leaf nodes contain the anchors. In T, a path from the root (i.e., metadata M) to a leaf node (i.e., an anchor) describes the position of the metadata with respect to the anchor. The easiest method to describe the position would be to count the steps from the metadata to the anchor.

```
1.  DFS (G, M) /* G is the ontology graph, M is the metadata */
2.  {
3.      List A = set of anchors in G
4.      List L = empty
5.      Tree T = empty
6.      Choose M as the starting vertex
7.      Search(M)
8.      While (L not empty)
9.      {
10.         Remove edge (v, w) from end of L
11.         If w not yet visited
12.         {
13.             Add edge(v, w) to T
14.             Search(w)
15.         }
16.     }
17. }
18.
19. Search(vertex v)
20. {
21.     Mark v as visited
22.     If v is an anchor in A
23.         return
24.     For each edge (v, w)
25.         Add edge (v, w) to end of L
26. }
```

The depth-first search algorithm is mainly for describing the metadata's position with respect to the anchors. When we build metrics describing the relative position between the metadata and non-anchor entities, we will still apply this algorithm to reference the metadata's position with respect to anchors. This is because the reasoning on the receiving side requires the information about anchors.

**Error! Reference source not found.** shows the pseudo code for matching on the receiving side. This width-first search algorithm takes the ontology graph G and the anchor (vertex of G) as input. It initializes a tree T to the starting vertex, and a list L which stores the edges that are visited (Line 3, 4). In the search function *Search(vertex v)*, the algorithm first marks the vertex as visited and assigns a score based on the metric description. If the score is smaller than a critical value, the search stops (Line 20, 21, 22, 23). If a vertex v has several unmarked neighbors, the edges between the vertex and the neighbors will be appended to the list L (Line 24, 25). As a width first search algorithm, it removes edges from start of list L so that the list acts as a queue rather than a stack (Line 9). Also, each vertex is clearly marked at most once, each edge is added to the list L at most once, and therefore removed from the list at most once. The tree T constructed by the algorithm is a width first search tree of the vertices reached during the search. These vertices represent the candidates that could potentially match the metadata.

```
1.  WFS (G, A) /* G is the ontology graph, A is the anchor */
2.  {
3.      List L = empty
4.      Tree T = empty
5.      Choose A as the starting vertex
6.      Search(A)
7.      While (L not empty)
8.      {
9.          Remove edge (v, w) from start of L
10.         If w not yet visited
11.         {
12.             Add edge(v, w) to T
13.             Search(w)
14.         }
15.     }
16. }
17.
18. Search(vertex v)
19. {
20.     Mark v as visited
21.     Assign a score S based on the metric description
22.     If the score S is smaller than a critical value
23.         return
24.     For each edge (v, w)
25.         Add edge (v, w) to end of L
26. }
```

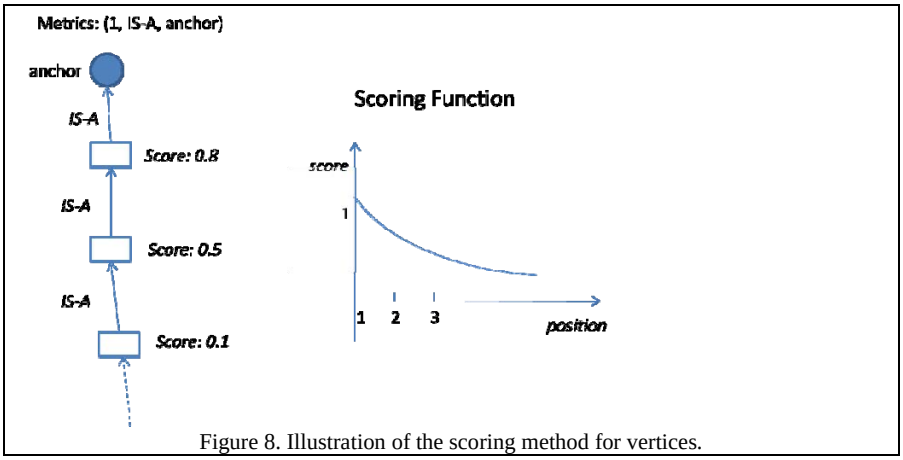

Figure 8. Illustration of the scoring method for vertices.

We recognize that how to assign score to a vertex is a research issue that needs further investigation. Figure 8 illustrates a simple example to generate the score for a vertex. The basic idea is to define a scoring function

based on the metric. According to the scoring function, each vertex will get a score based on its relative position to the anchor. The vertex that perfectly matches the metric gets a highest score; the next nearest vertex gets a smaller score, and so on. We will investigate various ways to define the scoring function.

## 5. Conclusions and Future Research

Our initial results indicate that very consistent matches can be achieved using the techniques describe in this paper. The quality of the matches between Ontologies is highly dependent on the choice of anchors, and the set of relationships (IS-A, etc…) and properties supported.

There are several points that require further research. Here we only list a few of them that we consider to be important:

- It is unclear how good anchor concepts are selected. There are certain requirements, such as them being sharable and general enough to facilitate successful ontology matching. What makes a good anchor is however unclear and will require evaluation of real world scenarios. A methodology to select good anchor points is needed.
- Once a data item has been successfully negotiated and transferred from domain A to domain B, it is important that the same data item can cross back, *without being incorrectly classified*. That is, a correct reverse operation must be guaranteed. The criteria for this operation must be defined, understood, and the ontology matching algorithms must guarantee this property.
- It is important that higher priority can be given to certain relationships. That is, in some cases it might be very important for the sending domain to ensure that a data item's metadata is matched against some concept in the receiving ontology that is closely related to some anchor *A* via the IS-A relationship. In contrast to, for example, the matching concept being the domain for a certain property. Hence, it should be possible for the sending domain to give *weight* to certain metrics that is sent to the receiving domain. The receiving domain must of course take this into consideration. How this weight is properly respected in the graph matching algorithm needs to be clarified.

## Acknowledgements

This research was partially funded by AFRL under contract# FA8750-09-C-0058.

## References

[1] Hu, W., Jian, N., Qu, Y., Wang, Y., "GMO: a graph matching for ontologies", in: Proceedings of K-CAP Workshop on Integrating Ontologies, 2005, pp. 41–48.

[2] Noy, N. and M. Musen, "Anchor-PROMPT: Using non-local context for semantic matching", Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle, WA, 2001

[3] Melnik, S., H. Garcia-Molina, et al., "Similarity flooding: a versatile graph matching algorithm and its application to schema matching", Proc. 18th International Conference on Data Engineering (ICDE), San Jose, CA, 2002.

[4] Qu, Y., Hu, W., Cheng, G, "Constructing virtual documents for ontology matching", in Proceedings of the 15th International World Wide Web Conference, ACM Press, 2006, pp. 23- 31, 2006

[5] Noy, N. and M. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment", in the Proc. of 17th AAAI, Austin, TX, 2000

[6] Doan, A., J. Madhaven, et al. (2003). "Learning to Match Ontologies on the Semantic Web." VLDB Journal 12(4): 303-319

[7] Mitra, P., N. Noy, et al., "OMEN: A Probabilistic Ontology Mapping Tool", Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC), 2004

[8] Hu, W., Qu, Y., Cheng, G., "Matching large ontologies: A divide-and-conquer approach", in Data & Knowledge Engineering, **vol. 67**, 2008, pp. 140-160.